CS 4803 / 7643: Deep Learning

Topics:

- Regularization
- Neural Networks

Dhruv Batra

Georgia Tech

Administrativia

- PS1/HW1 out
  - Available later today on Canvas
  - Due in 4 weeks
  - Asks about topics coming in the next couple of weeks
  - Please please please please start early
  - More details next class

Recap from last time

Parametric Approach: Linear Classifier

$\vec{s}$ $\quad \boxed{f(x,W)} = \boxed{W}\overset{3072\text{x}1}{\boxed{x}} + \boxed{b}$ 10x1

10x1 10x3072

Image

Array of **32x32x3** numbers
(3072 numbers total)

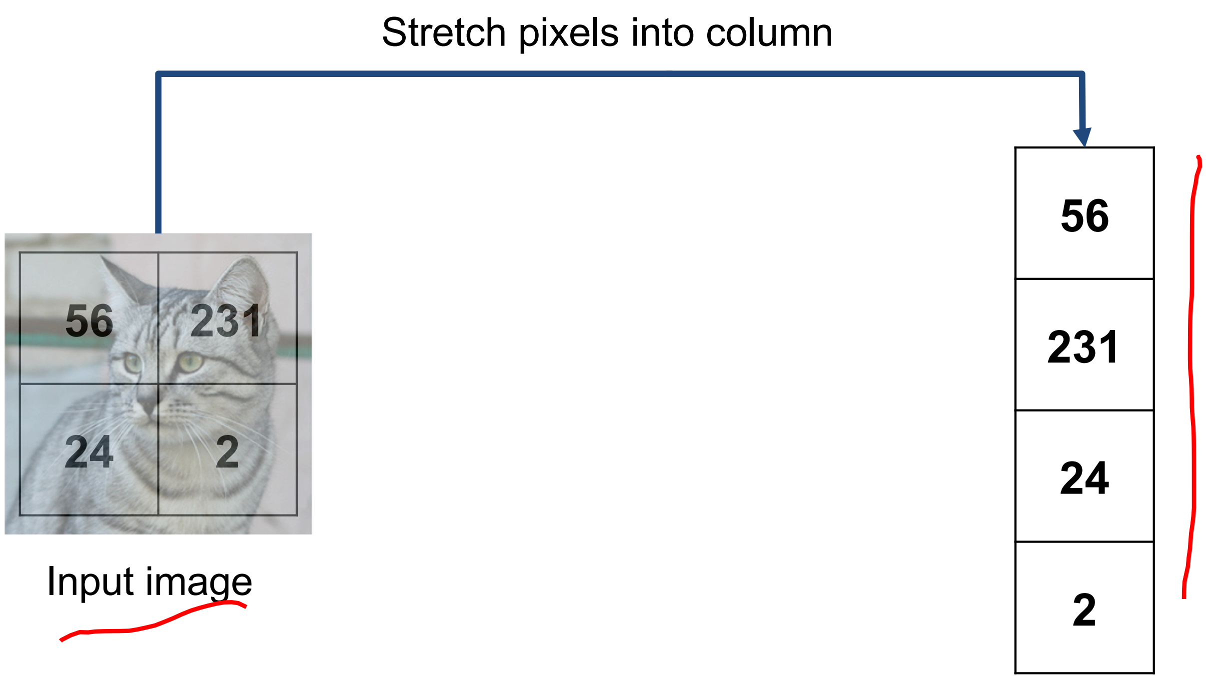
$\vec{x}$

$\rightarrow f(\mathbf{x},\mathbf{W}) \rightarrow$ **10** numbers giving class scores

W
parameters
or weights

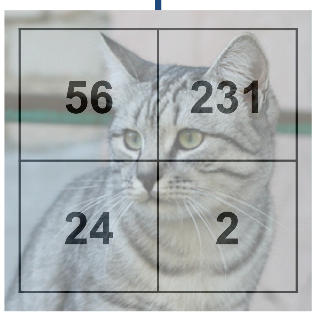
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



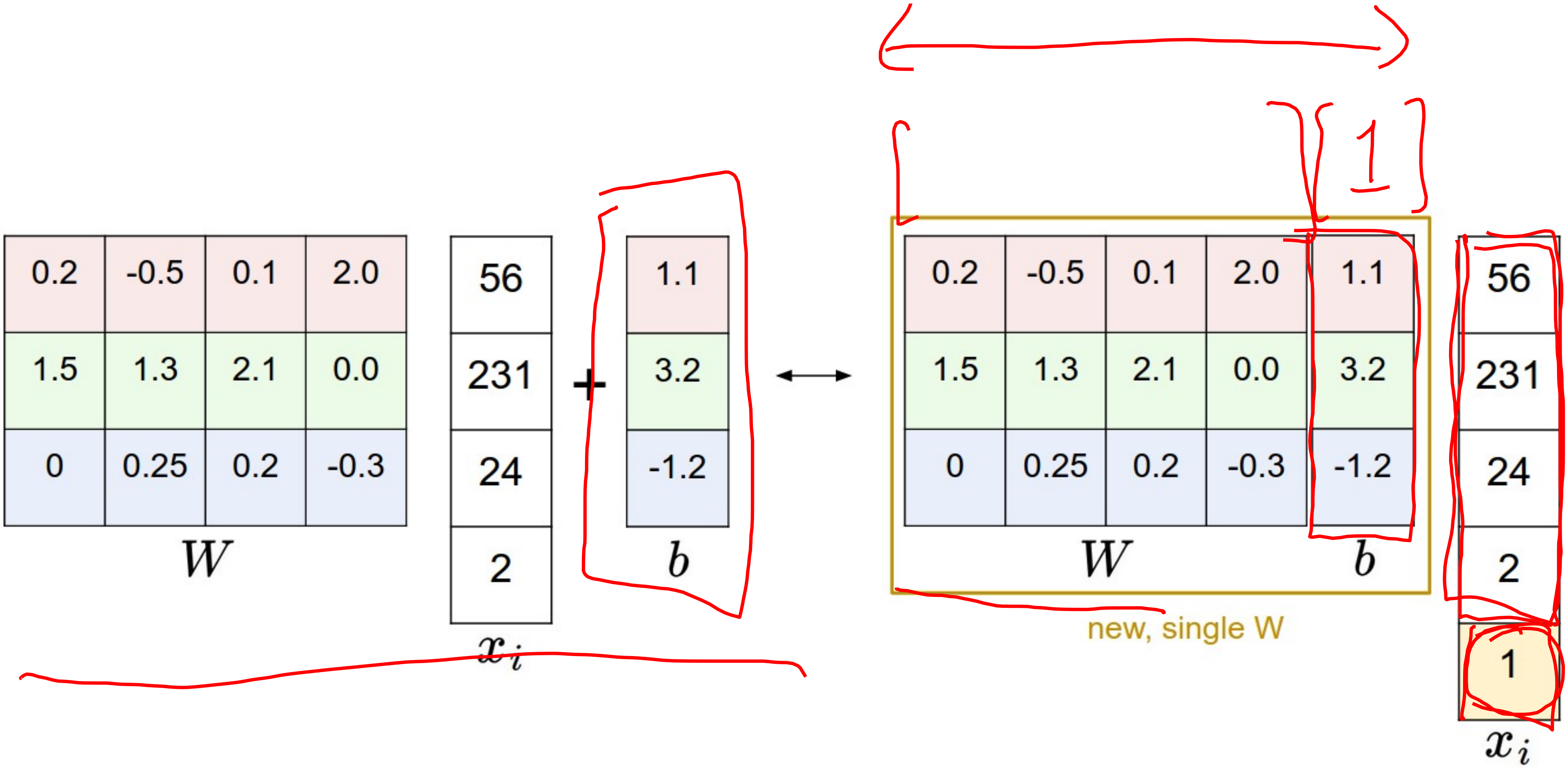
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Stretch pixels into column



Input image

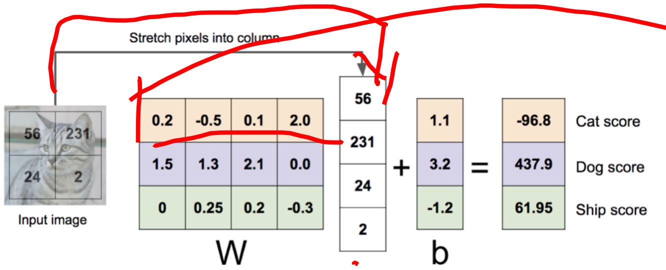
$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \begin{array}{l} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{array}$$


W
b
x_i
new, single W
1

Linear Classifier: Three Viewpoints

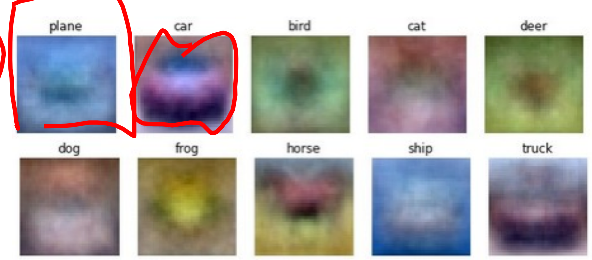
Algebraic Viewpoint

$$f(x,W) = Wx$$



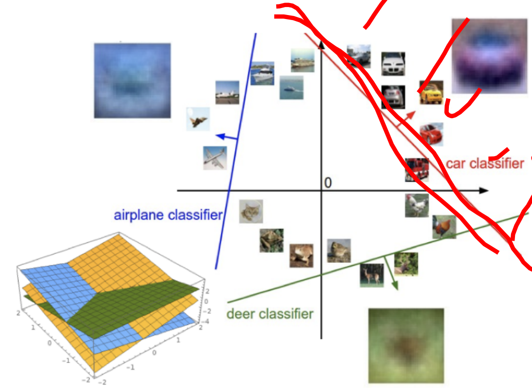
Visual Viewpoint

One template
per class



Geometric Viewpoint

Hyperplanes
cutting up space



Recall from last time: Linear Classifier

airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

1. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \qquad\qquad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

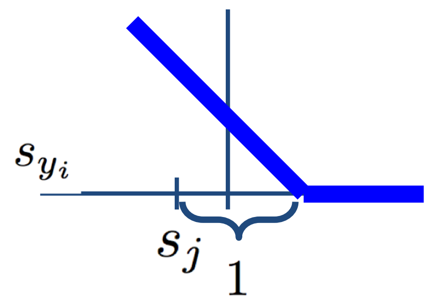
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

$W^* = [\,0\,]$

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$ are:

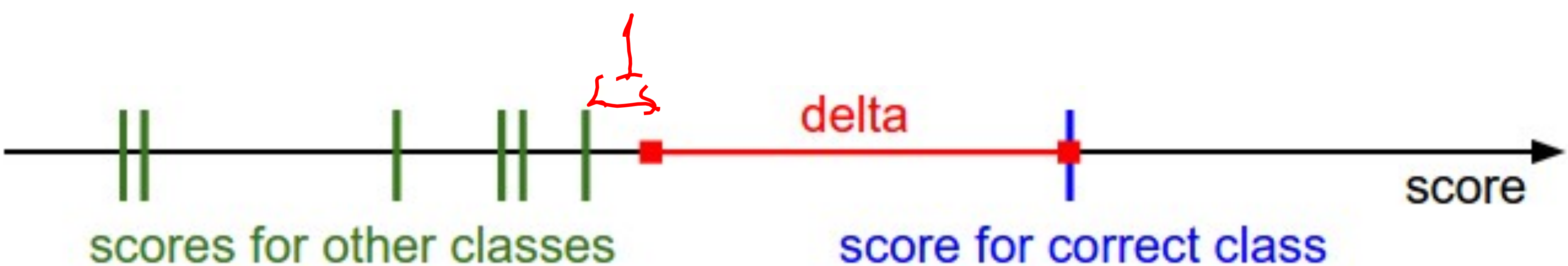
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Multiclass SVM loss:



$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + \cancel{1} & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, \underline{s_j - s_{y_i}} + \cancel{1})$$



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

cat	**3.2**
car	5.1
frog	-1.7

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

cat	**3.2**		**24.5**
car	5.1	exp →	164.0
frog	-1.7		0.18

unnormalized probabilities

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

			Probabilities must be >= 0		Probabilities must sum to 1
cat	**3.2**		**24.5**		**0.13**
car	5.1	$\xrightarrow{\text{exp}}$	164.0	$\xrightarrow{\text{normalize}}$	0.87
frog	-1.7		0.18		0.00
			unnormalized probabilities		probabilities

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = \boxed{y_i} | X = x_i)$$

	Unnormalized log-probabilities / logits	$\xrightarrow{\text{exp}}$ unnormalized probabilities	$\xrightarrow{\text{normalize}}$ probabilities
cat	**3.2**	**24.5**	**0.13**
car	5.1	164.0	0.87
frog	-1.7	0.18	0.00

→ $L_i = -\log(0.13)$
= **2.04**

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

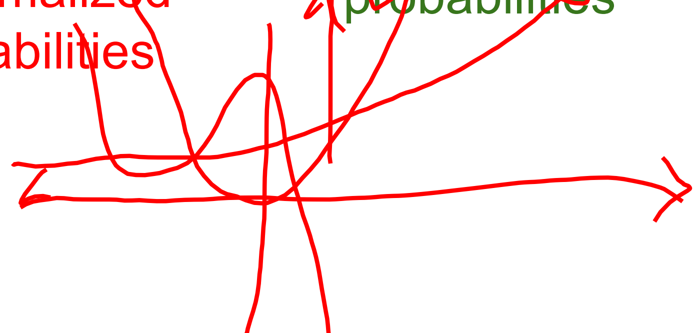
Softmax Function

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0 unnormalized probabilities	normalize	Probabilities must sum to 1 probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

$\rightarrow$ $L_i = -\log(0.13)$
$= \mathbf{2.04}$

Maximum Likelihood Estimation
Choose probabilities to maximize the likelihood of the observed data



Log-Likelihood / KL-Divergence / Cross-Entropy

$D = \{(x_i, y_i)\}$ IID $\sim P^*$

$$\boxed{\hat{w}_{MLE}} = \max P(D \mid w)$$

$$= \operatorname{argmax} \log P(D \mid w)$$

$$= \max \sum_i \log P(y_i \mid x_i, w)$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

$$L_i = -\log P(Y = y_i | X = x_i)$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	$\xrightarrow{\text{exp}}$	unnormalized probabilities	$\xrightarrow{\text{normalize}}$	probabilities	compare	Correct probs
cat	**3.2**		**24.5**		**0.13**	→ compare ←	**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits		unnormalized probabilities (Probabilities must be >= 0)		probabilities (Probabilities must sum to 1)		Correct probs
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**	→ compare ←	**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Kullback–Leibler divergence

$$D_{KL}(P \| Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits	$\xrightarrow{\text{exp}}$	unnormalized probabilities (Probabilities must be >= 0)	$\xrightarrow{\text{normalize}}$	probabilities (Probabilities must sum to 1)		Correct probs
cat	**3.2**		**24.5**		**0.13**	→ compare ←	**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross Entropy

$$H(P, Q) = H(p) + D_{KL}(P \| Q)$$

1 -

1

0

0

$e^{s_3} \rightarrow 0$

$\Sigma(\ )$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Plan for Today

- (Finish) Loss Functions
- Regularization
- Neural Networks

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ Softmax Function

cat	**3.2**
car	5.1
frog	-1.7

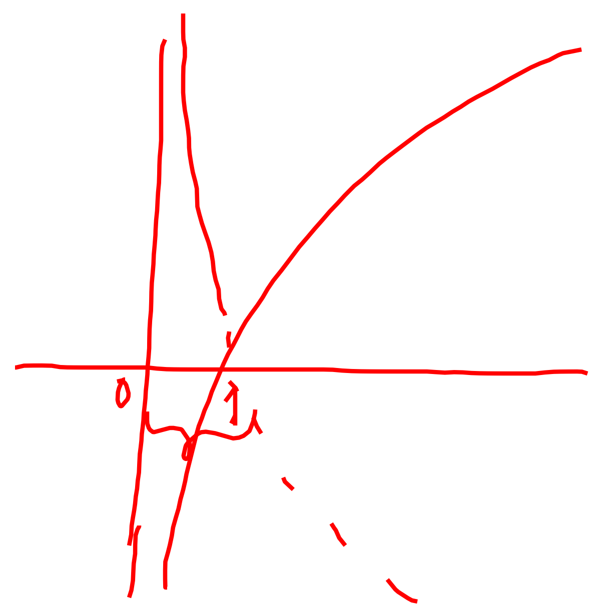
Maximize probability of correct class

$$L_i = -\log P(Y = y_i|X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Q: What is the min/max possible loss L_i?



x_i

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

cat	**3.2**
car	5.1
frog	-1.7

Maximize probability of correct class

$$L_i = -\log P(Y = y_i|X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Q: What is the min/max possible loss L_i?
A: min 0, max infinity

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

cat	$\pm\varepsilon$ 0	**3.2**
car	$\pm\varepsilon$ 0	5.1
frog	$\pm\varepsilon$ 0	-1.7

Maximize probability of correct class

$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{\boxed{e^{s_{y_i}}}}{\sum_j \boxed{e^{s_j}}}\right)$$

Q2: At initialization all s will be approximately equal; what is the loss?

$$-\log\left(\frac{1}{K}\right)$$

$$= \log K$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

Maximize probability of correct class

$$L_i = -\log P(Y = y_i|X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

cat	**3.2**
car	5.1
frog	-1.7

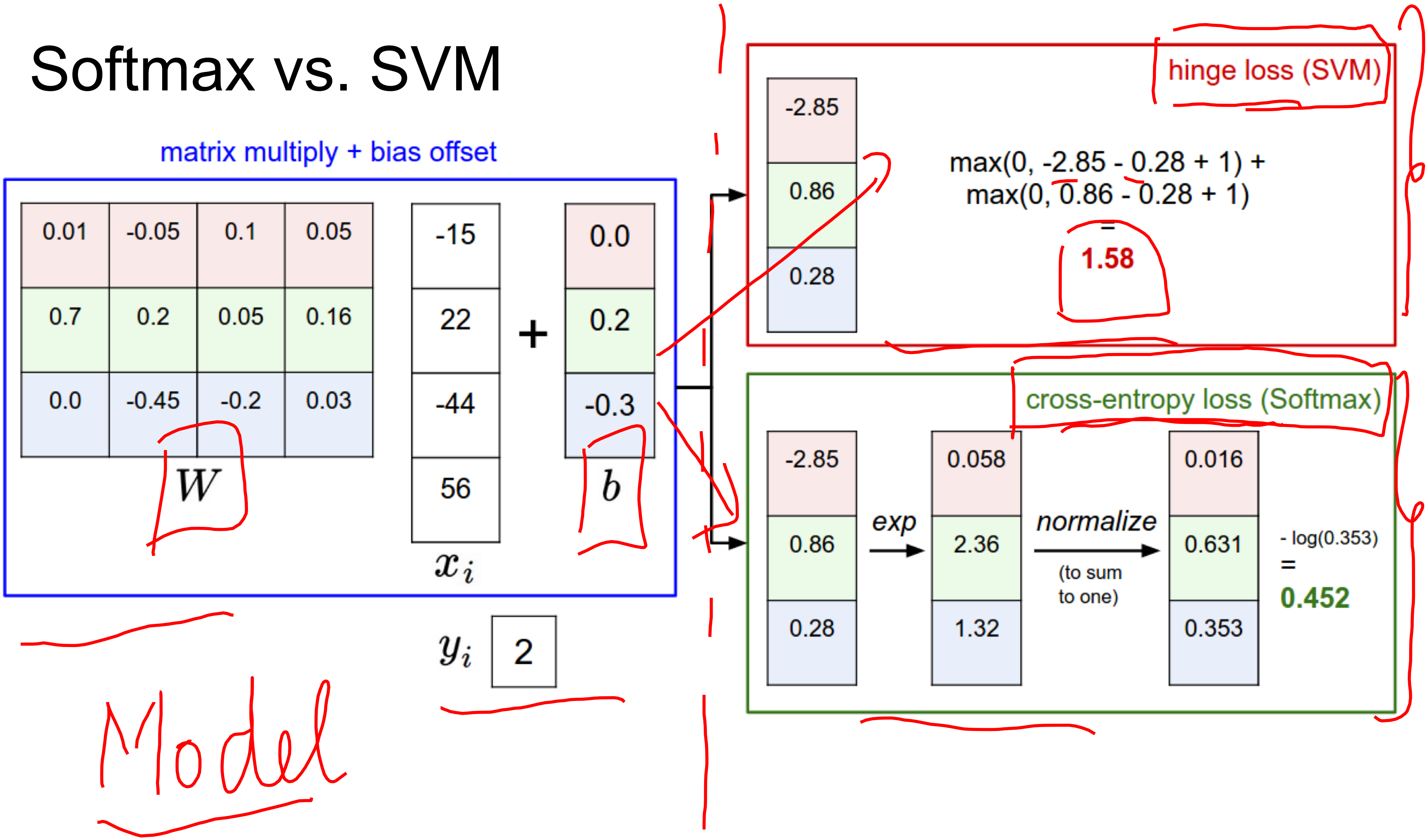
Q2: At initialization all s will be approximately equal; what is the loss?
A: log(C), eg log(10) ≈ 2.3

Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Softmax vs. SVM



Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}+\varepsilon}}{\sum_j e^{s_j+\varepsilon}}\right) \quad \text{vs} \quad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3] ± [ε ε ε]

[10, 9, 9]

[10, -100, -100]

and $\boxed{y_i = 0}$

Q: Suppose I take a datapoint and I jiggle a bit (changing its score slightly). What happens to the loss in both cases?

Plan for Today

- (Finish) Loss Functions
- Regularization
- Neural Networks

Regularization

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data

Regularization

MLE

$\log P(\underline{W})$

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Regularization

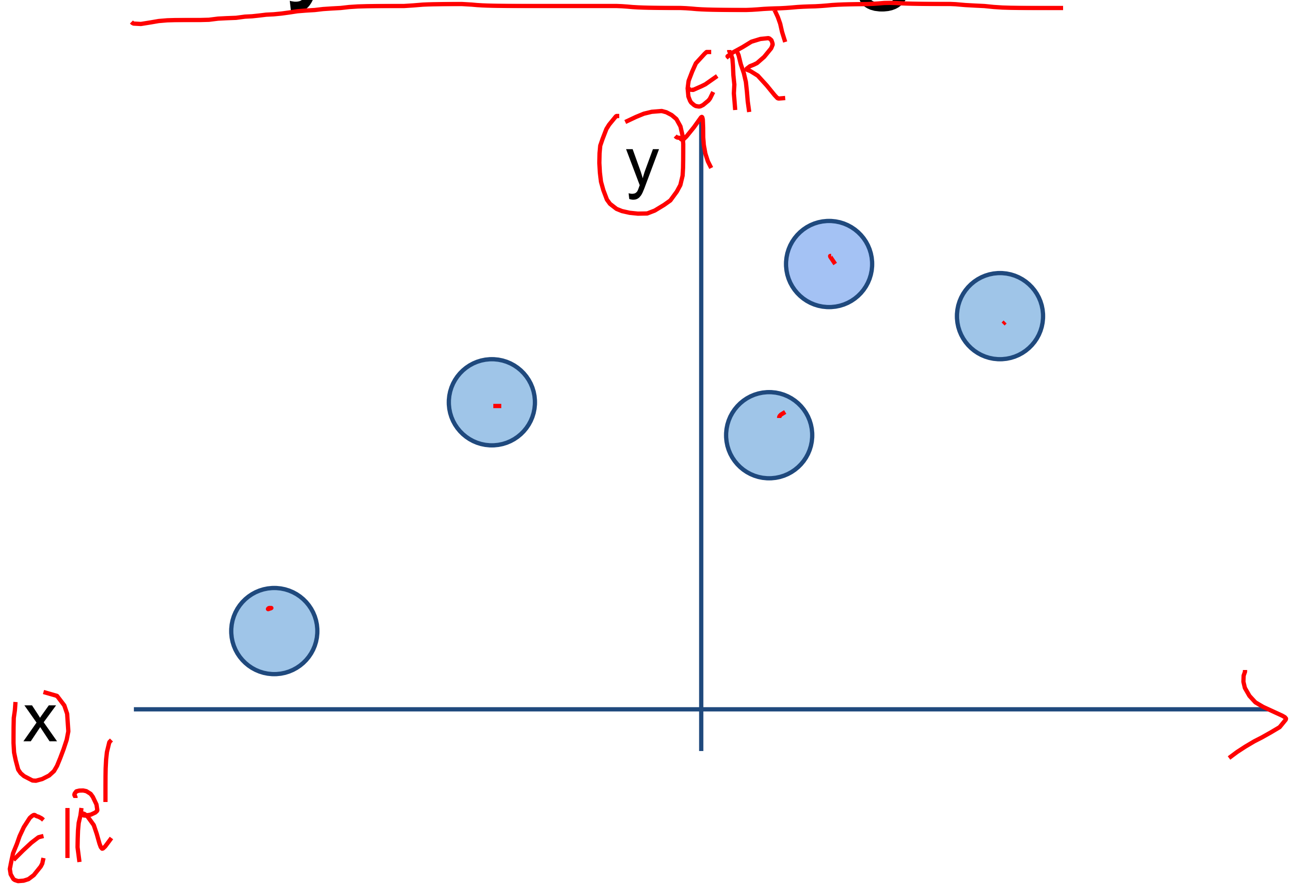
λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

Data loss: Model predictions should match training data

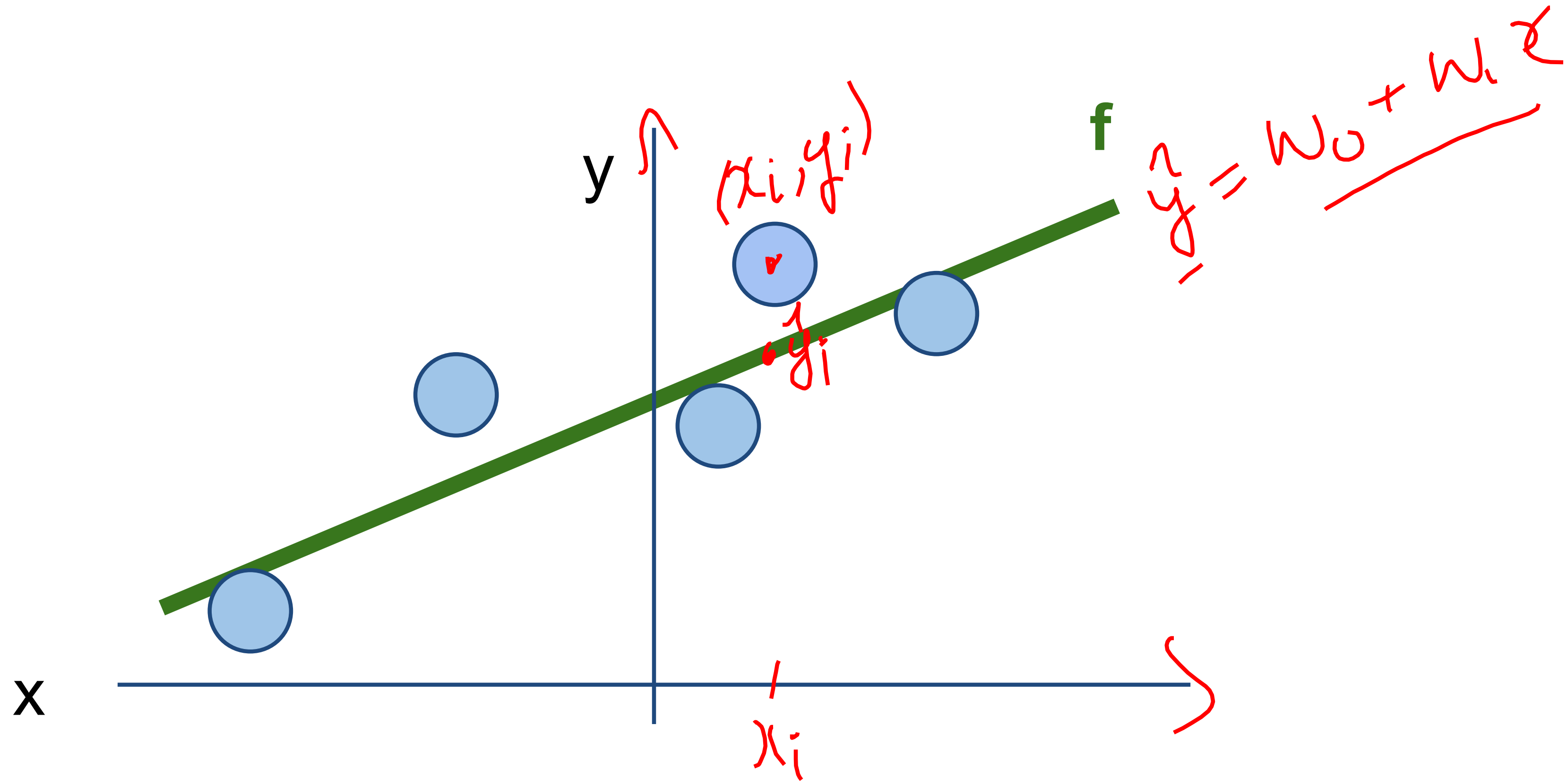
Regularization: Prevent the model from doing *too* well on training data

Regularization Intuition in Polynomial Regression



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Polynomial Regression



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Polynomial Regression

$$\hat{y} = w_0 + w_1 x \qquad 1$$

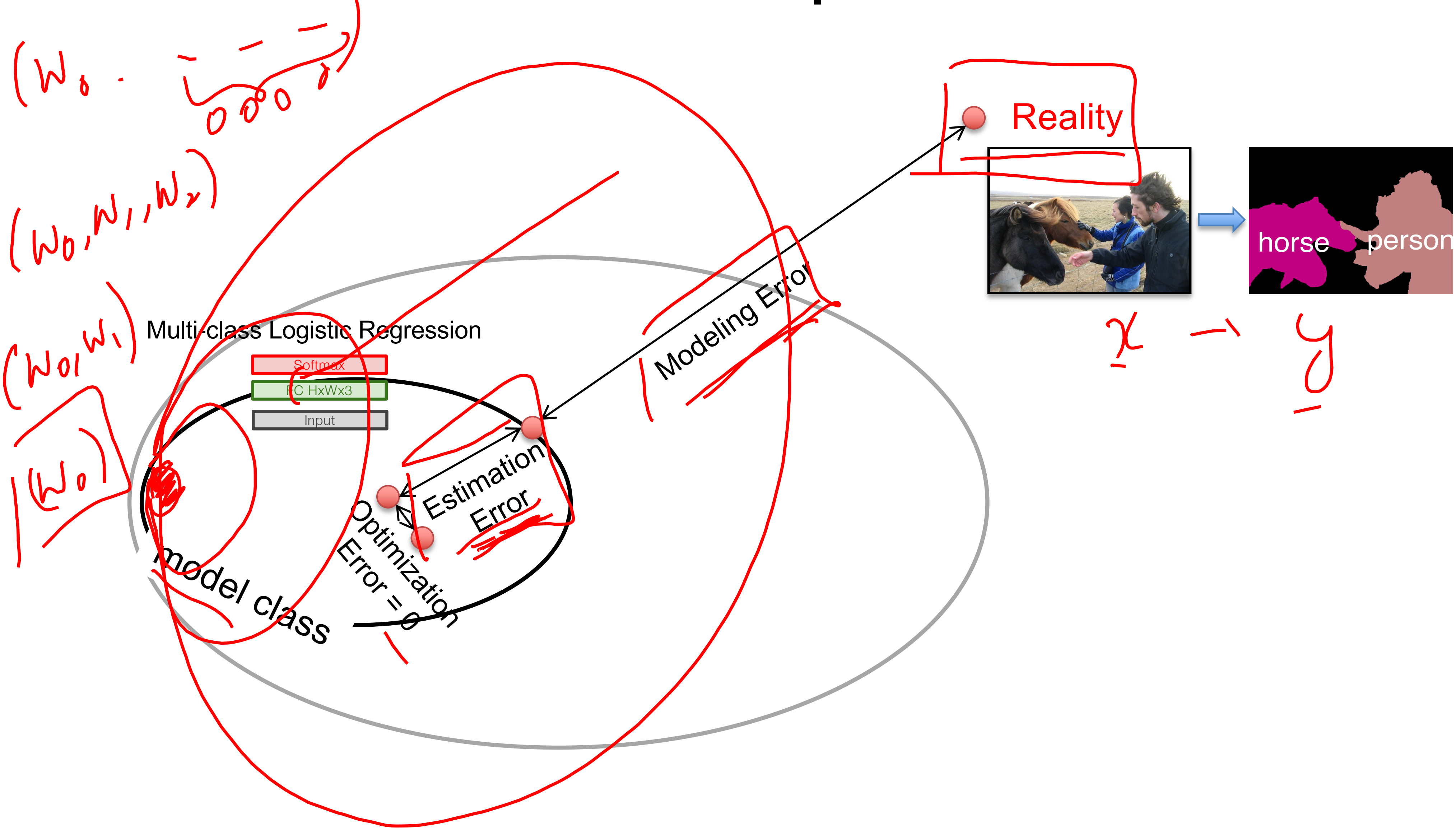
$$= w_0 + w_1 x + w_2 x^2$$

$$= w_0 + \cdots\cdots + w_d x^d \qquad d\text{-degree}$$

$$= [w_0 \; \ldots\ldots \; w_d] \underbrace{\begin{bmatrix} 1 \\ x^1 \\ x^2 \\ \vdots \\ x^d \end{bmatrix}}_{} = \vec{w}^T \vec{\phi}(\underset{\sim}{x})$$

Polynomial Regression

Error Decomposition



Polynomial Regression

$y = f(x)$

- Demo: https://arachnoid.com/polysolve/

- You are a scientist studying runners.
  – You measure average speeds of the best runners at different ages.

- Data: Age (years), Speed (mph)
  – 10 6
  – 15 9
  – 20 11
  – 25 12
  – 29 13
  – 40 11
  – 50 10
  – 60 9

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Regularization

λ = regularization strength (hyperparameter)

$$\boxed{L(W)} = \underbrace{\boxed{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}} + \underbrace{\boxed{\lambda R(W)}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

L2 regularization: $R(W) = \boxed{\sum_k \sum_l} W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\boxed{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}} + \underbrace{\cancel{\lambda R(W)}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

<u>L2 regularization:</u> $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Why regularize?

- Express preferences over weights
- Make the model *simple* so it works on test data
- Improve optimization by adding curvature

Recap

- We have some dataset of (x,y)
- We have a **score function:** $s = f(x; W) \overset{\text{e.g}}{=} \boxed{Wx}$
- We have a **loss function**:

Softmax

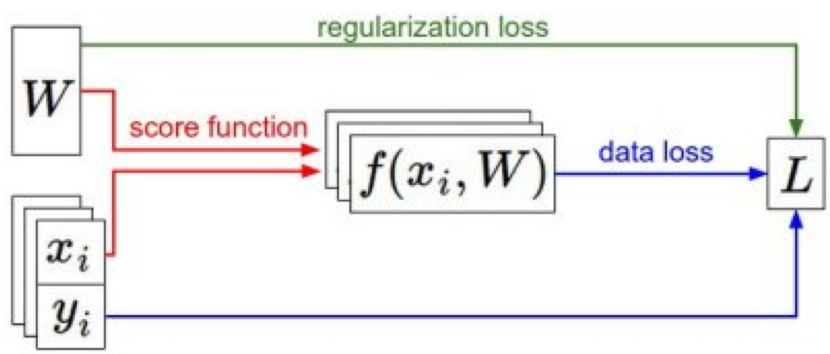
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

SVM

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \boxed{R(W)}$$

Full loss



Recap

How do we find the best W?

- We have some dataset of (x,y)
- We have a **score function:** $s = f(x; W) \stackrel{\text{e.g.}}{=} Wx$
- We have a **loss function**:

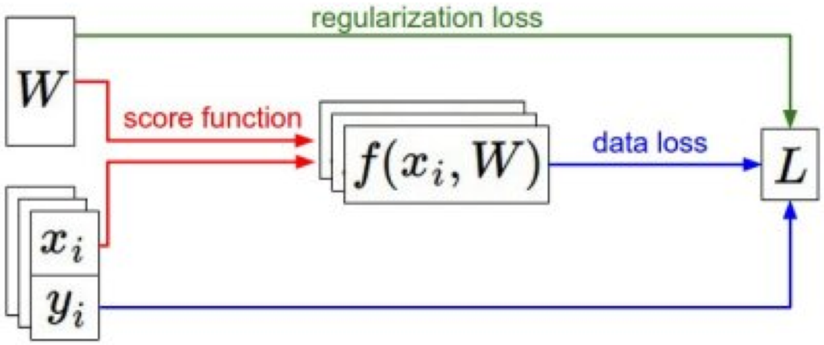
Softmax

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

SVM

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W)$$ Full loss



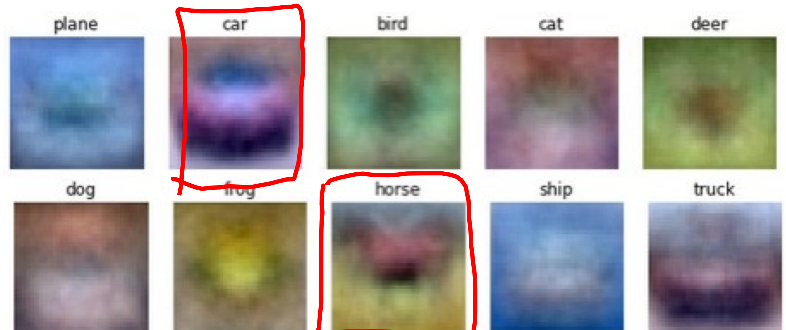
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Next: Neural Networks

So far: Linear Classifiers

x

$$f(x) = \boxed{W}x$$



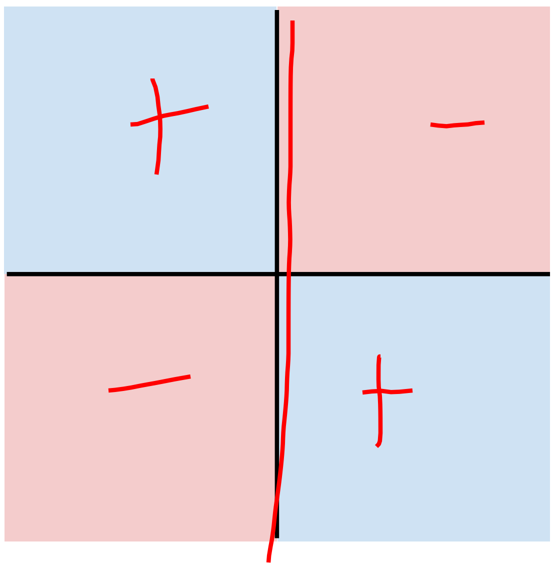
Class scores

s

Hard cases for a linear classifier

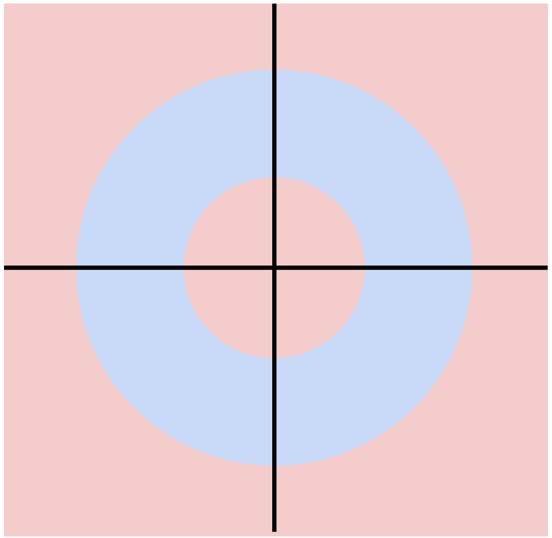
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



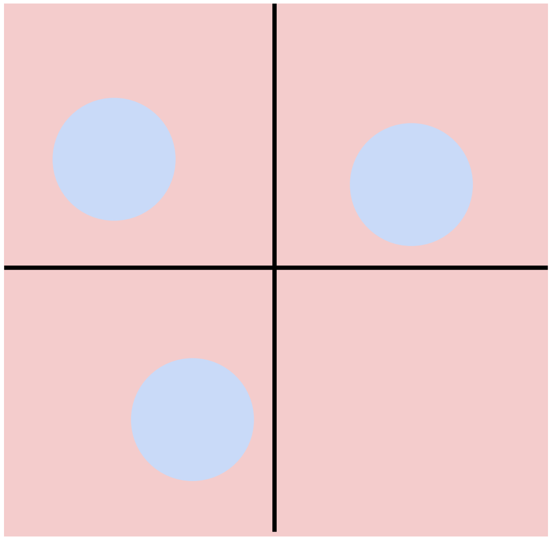
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Aside: Image Features

I

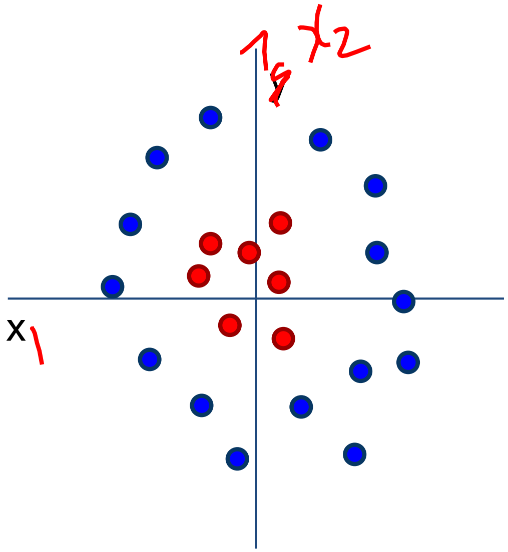
Raw Input

$f(x) = Wx$

Feature Representation

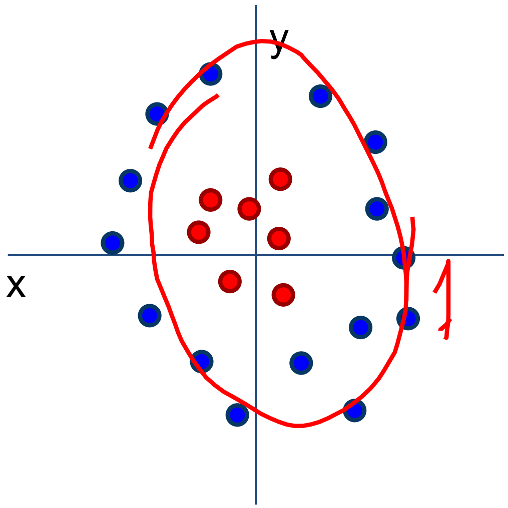
Class scores

Image Features: Motivation



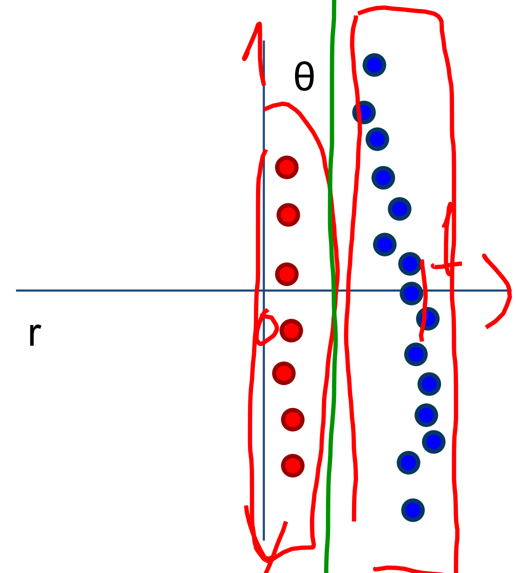
Cannot separate red and blue points with linear classifier

Image Features: Motivation



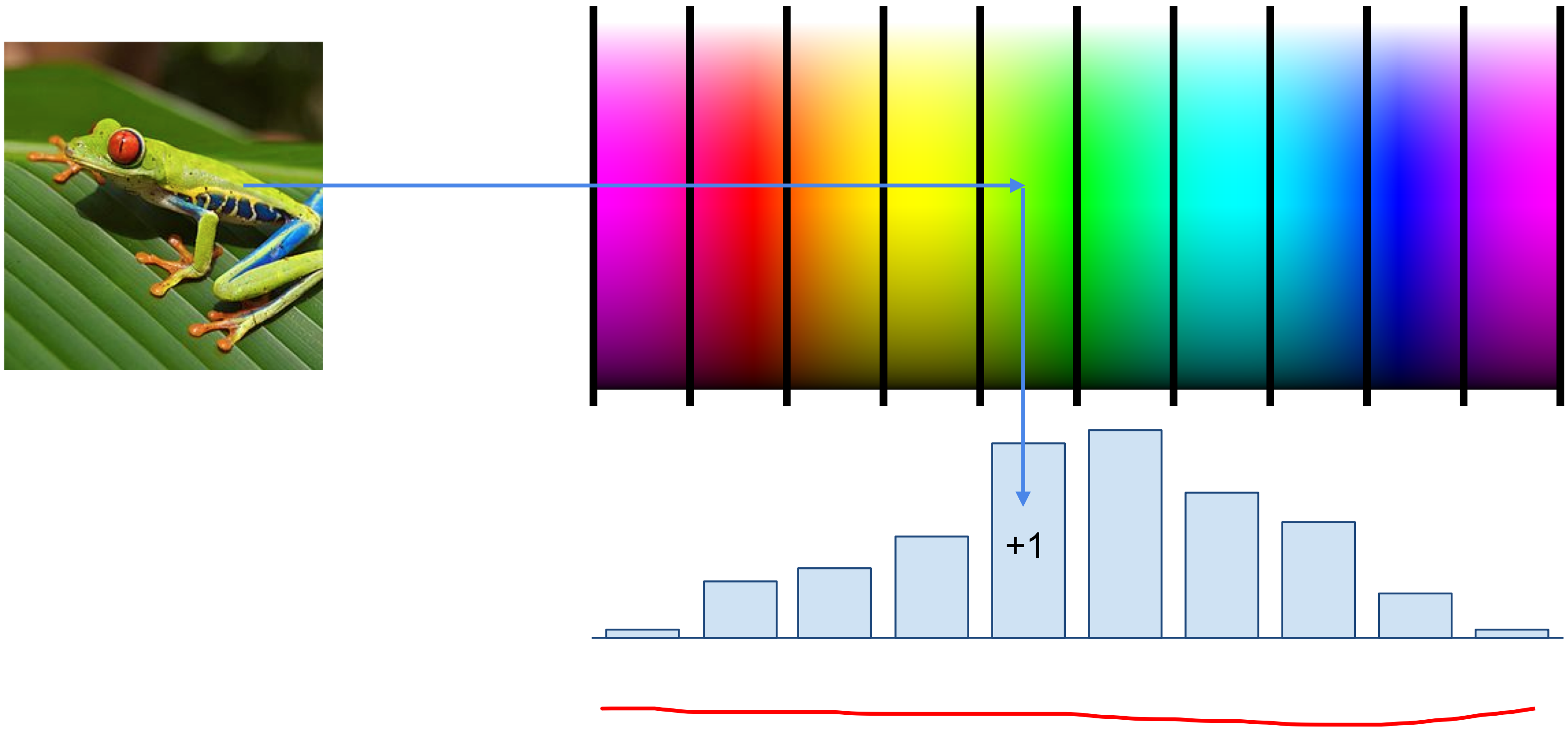
Cannot separate red and blue points with linear classifier

$$f(x, y) = (r(x, y), \theta(x, y))$$



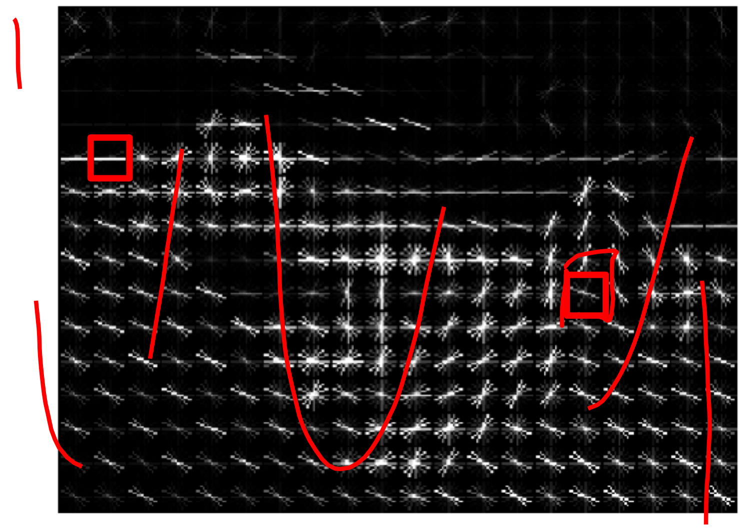
After applying feature transform, points can be separated by linear classifier

Example: Color Histogram



Example: Histogram of Oriented Gradients (HoG)

Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins

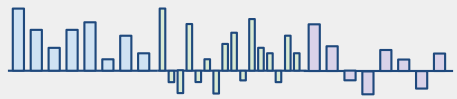


Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Lowe, “Object recognition from local scale-invariant features”, ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

Image features vs Neural Nets

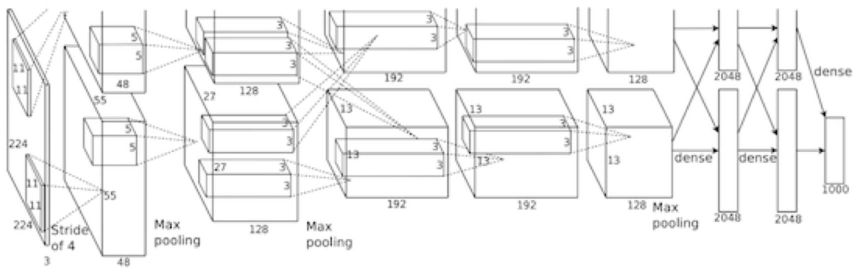
Feature Extraction



f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, “Imagenet classification with deep convolutional neural networks”, NIPS 2012.
Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

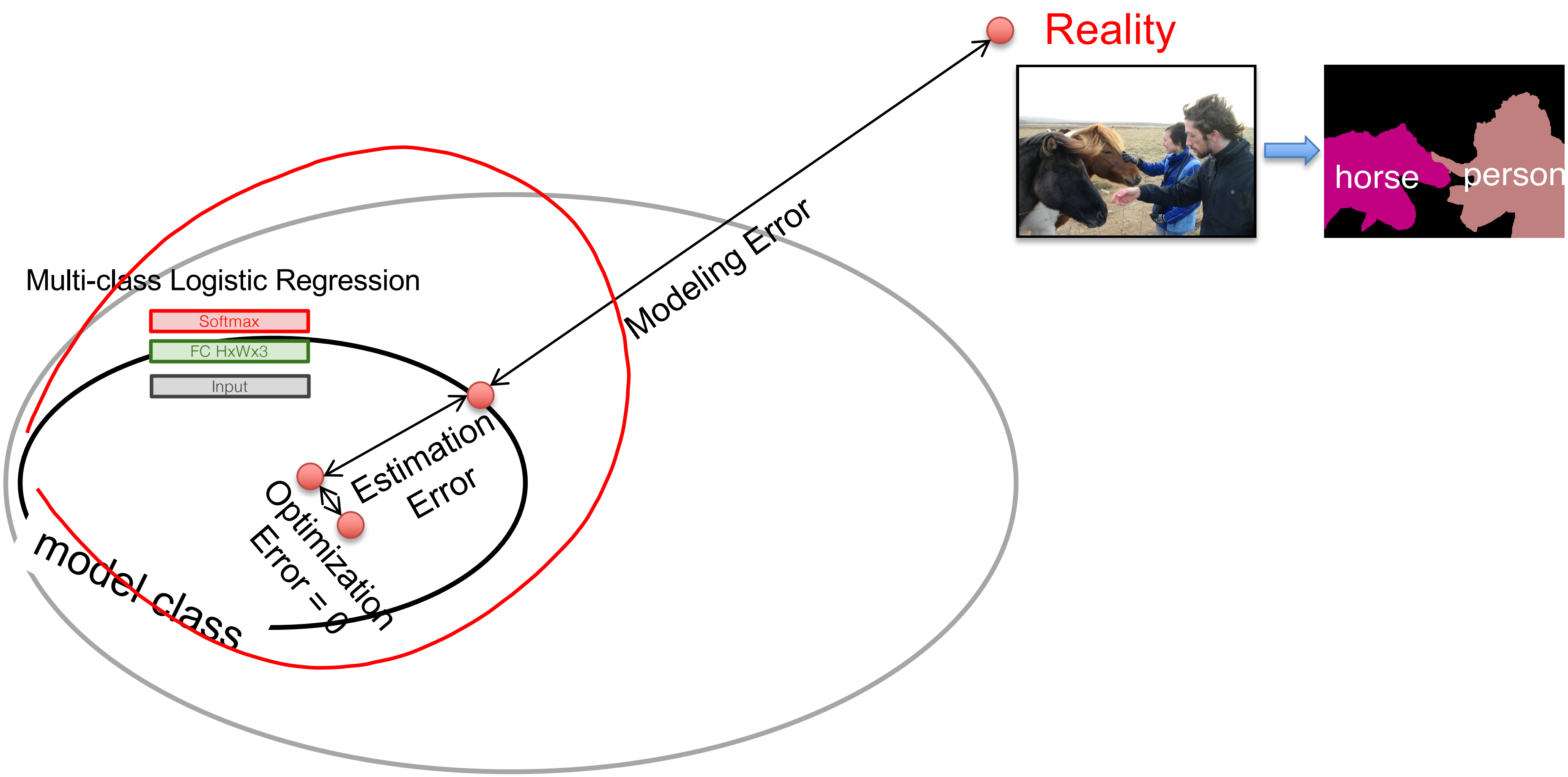
10 numbers giving scores for classes

training

Raw Input x

s

Error Decomposition



Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

$= \underbrace{W^{(2)} W^{(1)}}_{W^{(3)}} \vec{x}$

Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

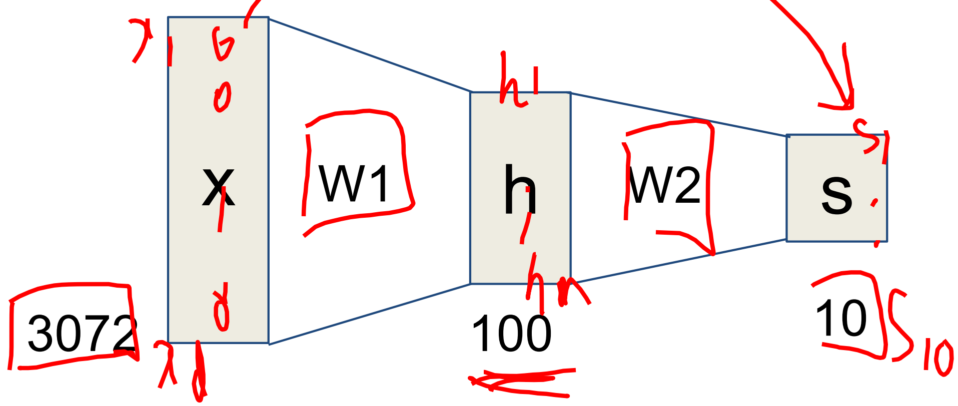
$W^{(3)} x$

Neural networks: without the brain stuff

$\vec{h}^{(1)}$

(**Before**) Linear score function: $f = Wx$

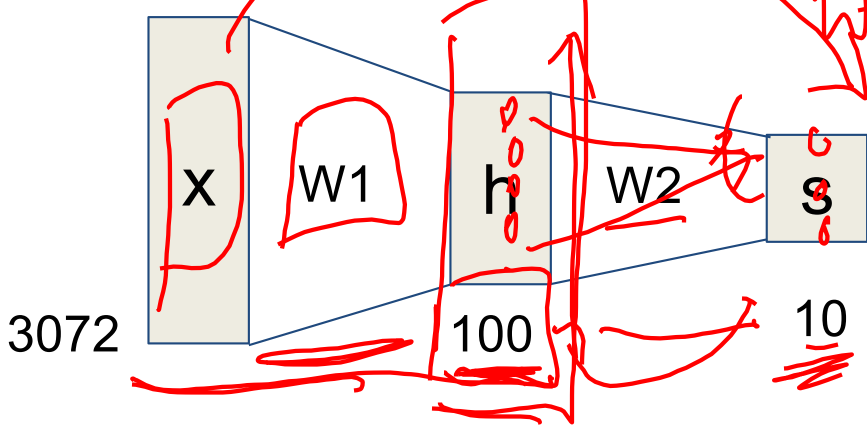
(**Now**) 2-layer Neural Network $f = W_2 \boxed{\max(0, W_1 \vec{x})}$



Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Neural networks: without the brain stuff

$\sigma(a) = \frac{1}{1+e^{-a}}$

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

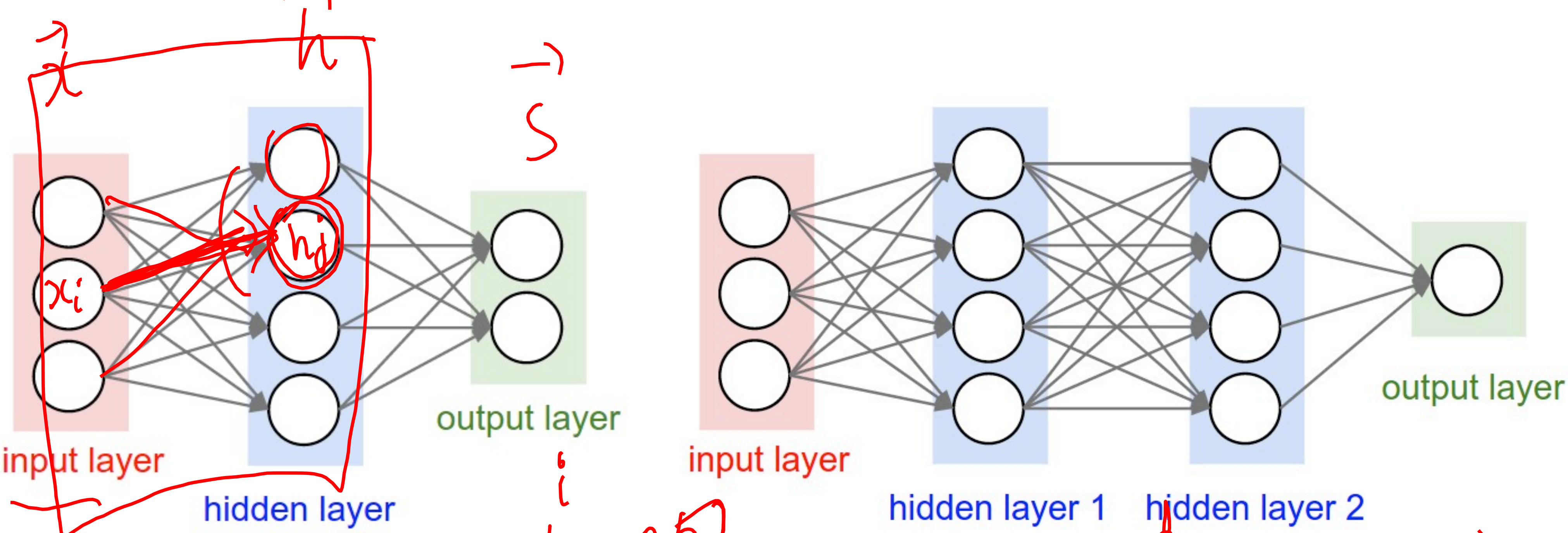
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Non-linearity

Multilayer Networks

- Cascaded "neurons"
- The output from one layer is the input to the next
- Each layer has its own sets of weights



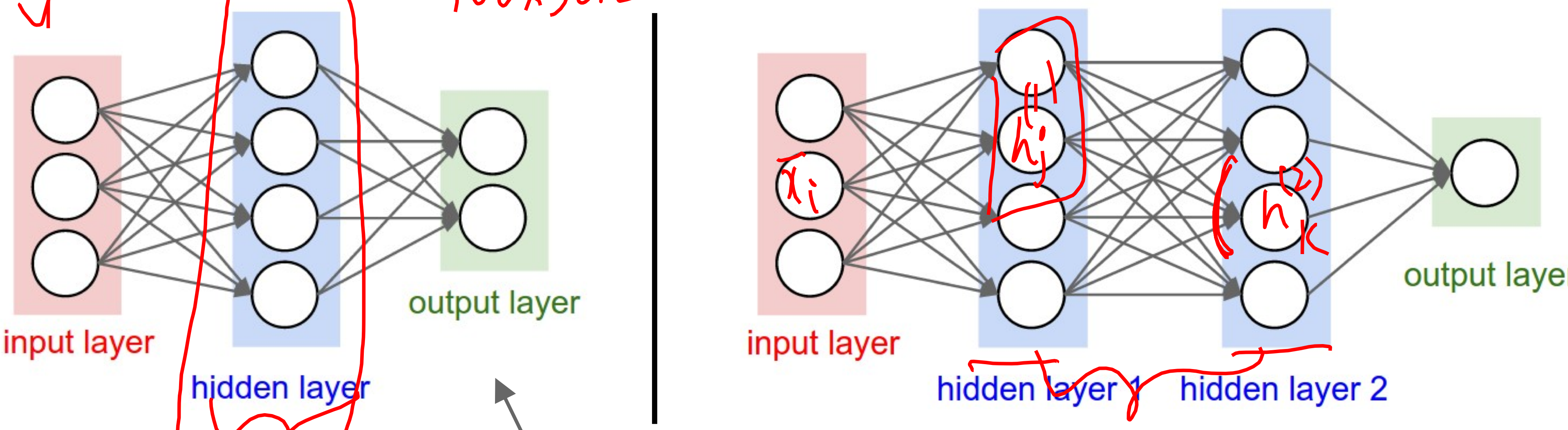
$W^{(1)}$

$w^{(1)}_{ji}$

$$h_j = \left(\sum_{i=1}^{d} w^{(1)}_{ji} x_i\right)$$

Neural networks: Architectures

$W^{(1)} = [\quad]$ 100x3072

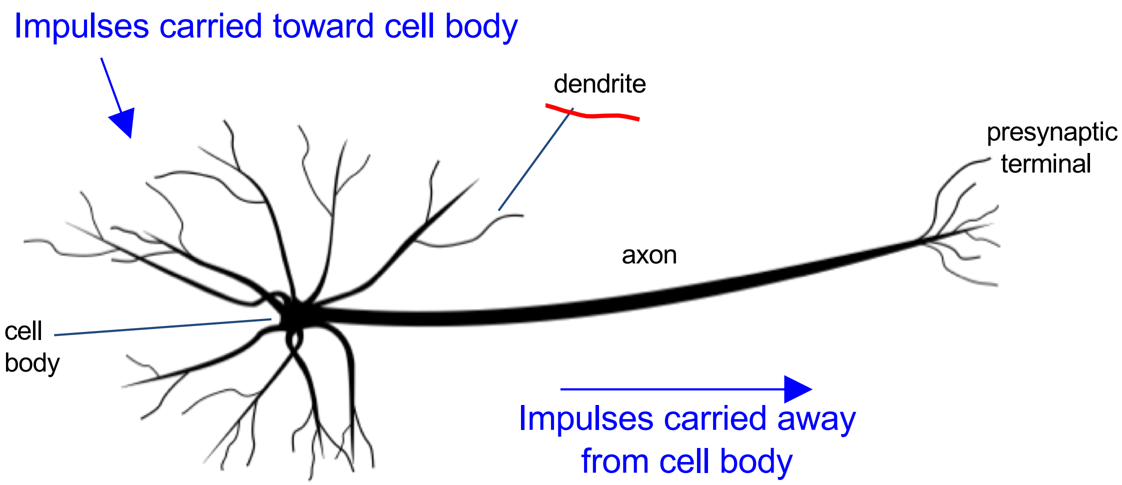


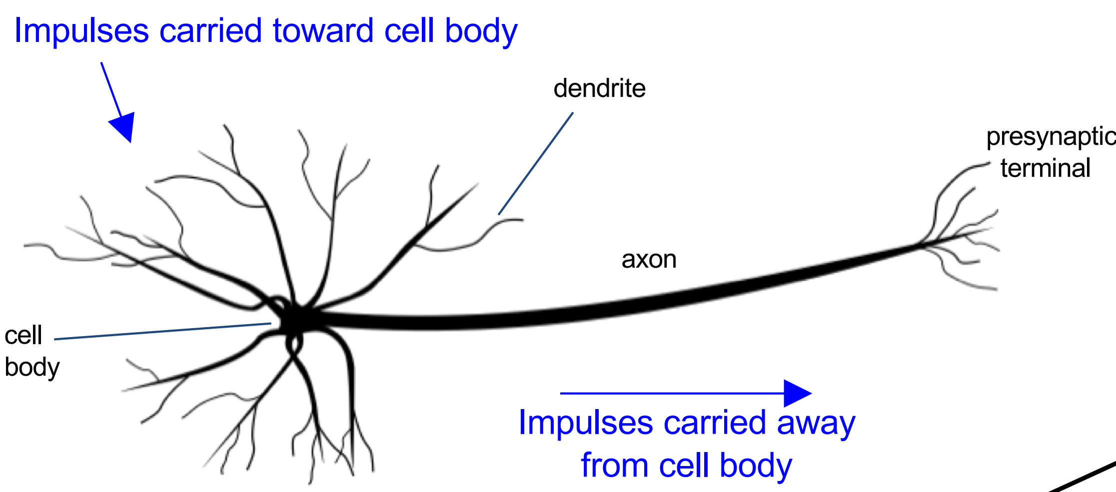
"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"Fully-connected" layers

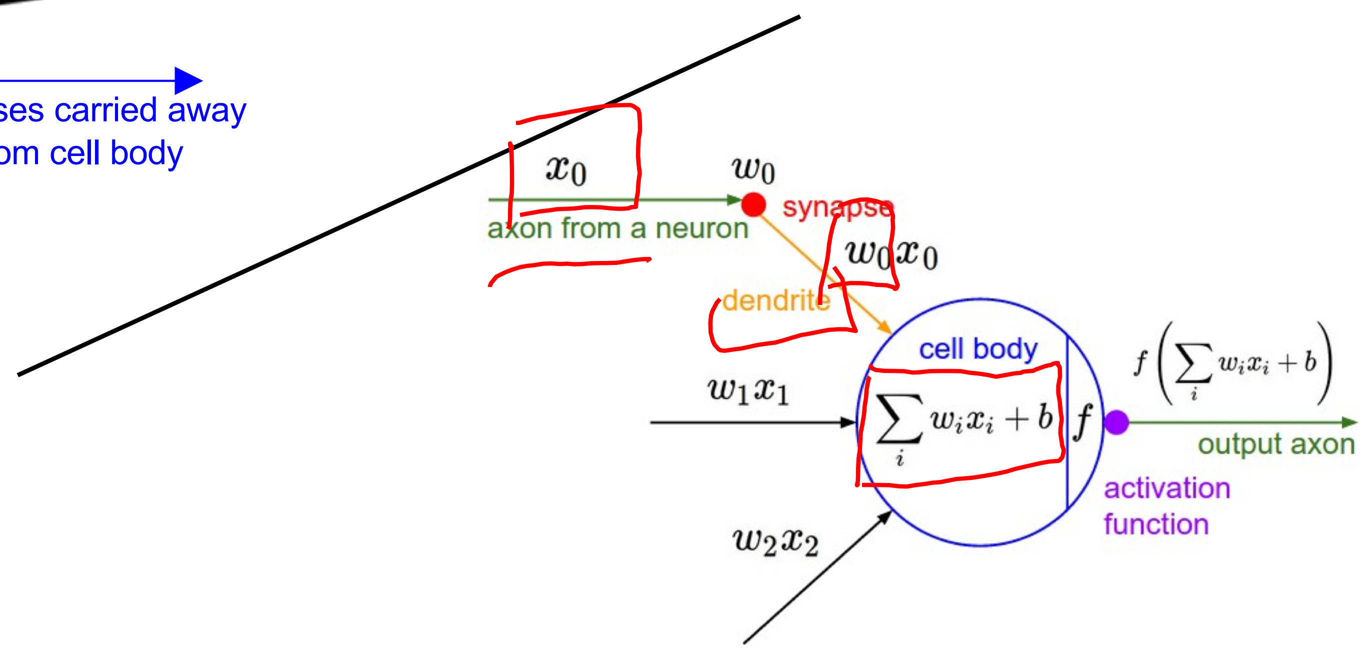
"3-layer Neural Net", or
"2-hidden-layer Neural Net"

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

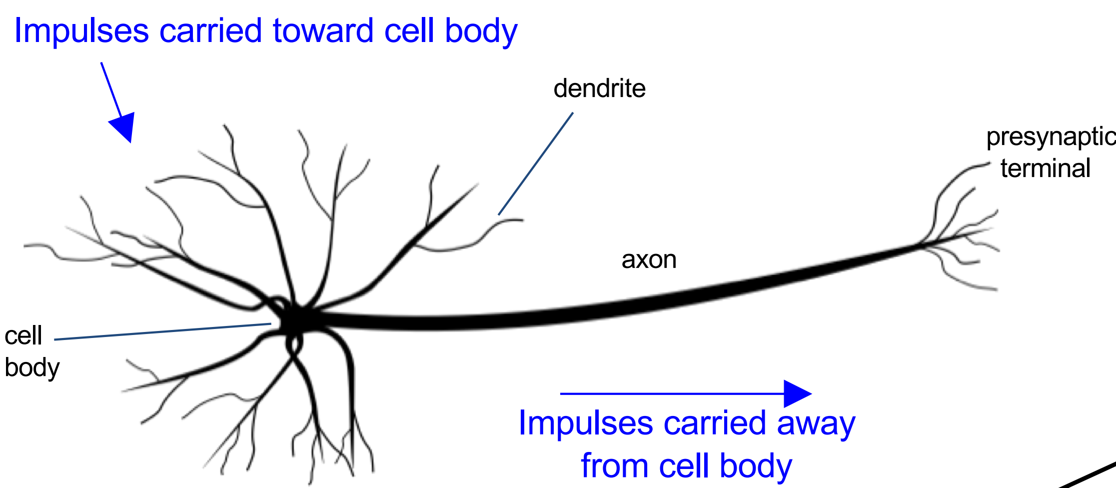

Impulses carried toward cell body
dendrite
presynaptic
terminal
axon
cell
body
Impulses carried away
from cell body

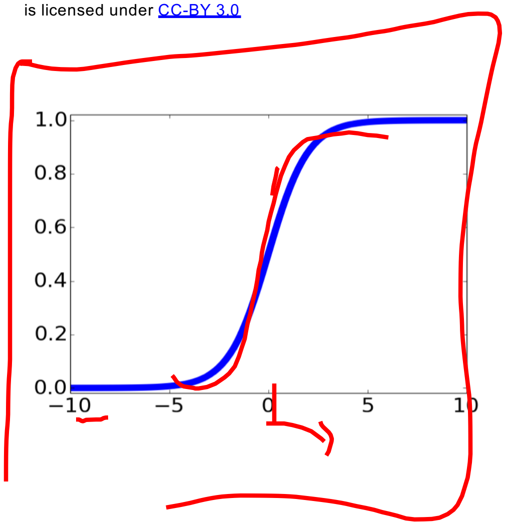


Impulses carried toward cell body
dendrite
presynaptic terminal
axon
cell body
Impulses carried away from cell body



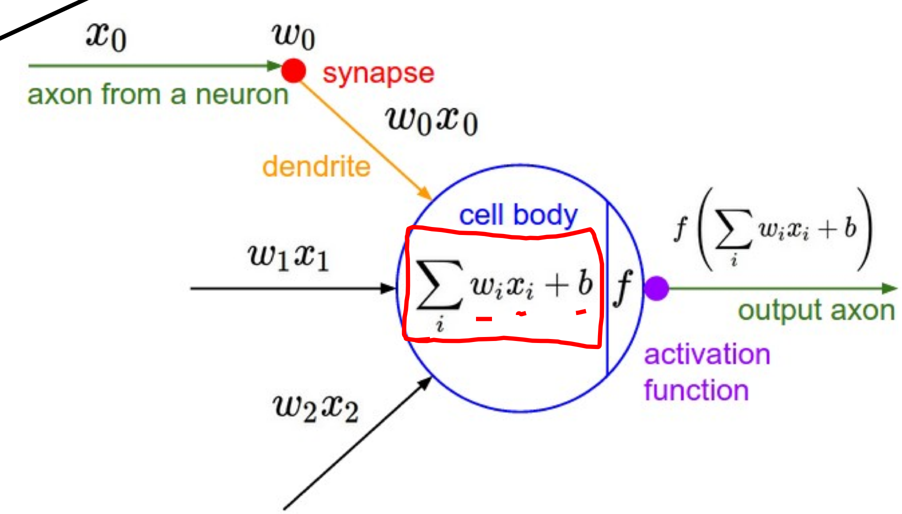
x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation function
$w_2 x_2$





sigmoid activation function

$$\frac{1}{1+e^{-x}}$$



Be very careful with your brain analogies!

Biological Neurons:

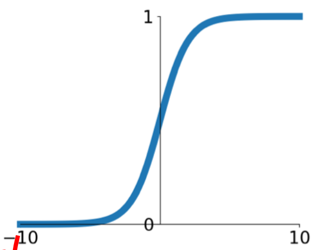
- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate

[Dendritic Computation. London and Hausser]

Activation functions

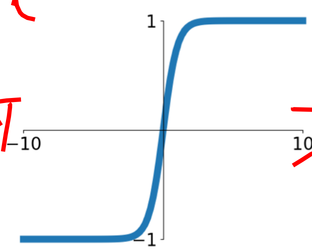
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



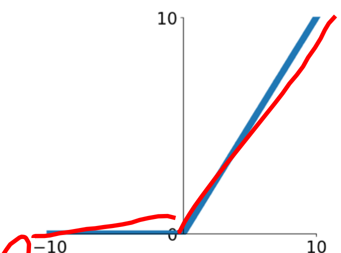
tanh

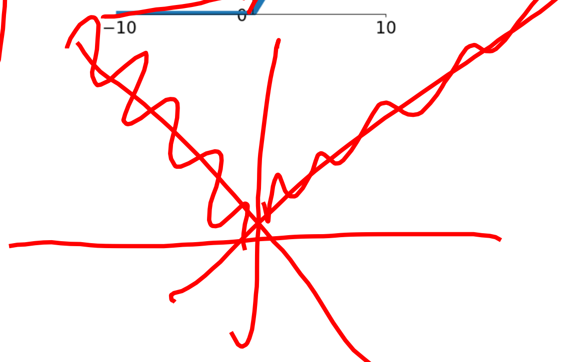
$\tanh(x)$ $= \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = 2\sigma(2x) - 1$



ReLU

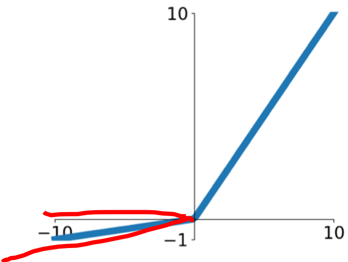
$\max(0, x)$





Leaky ReLU

$\max(0.1x, x)$

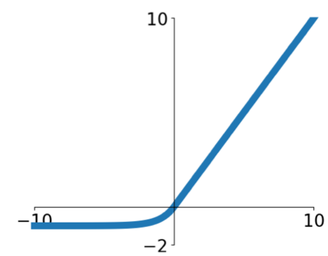


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



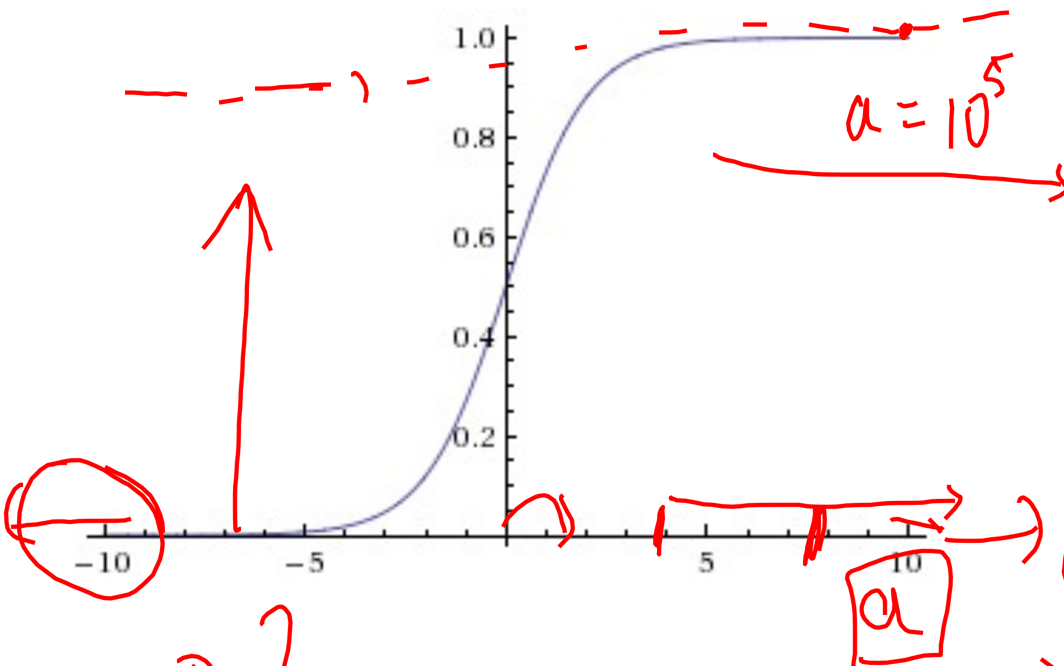
Activation Functions

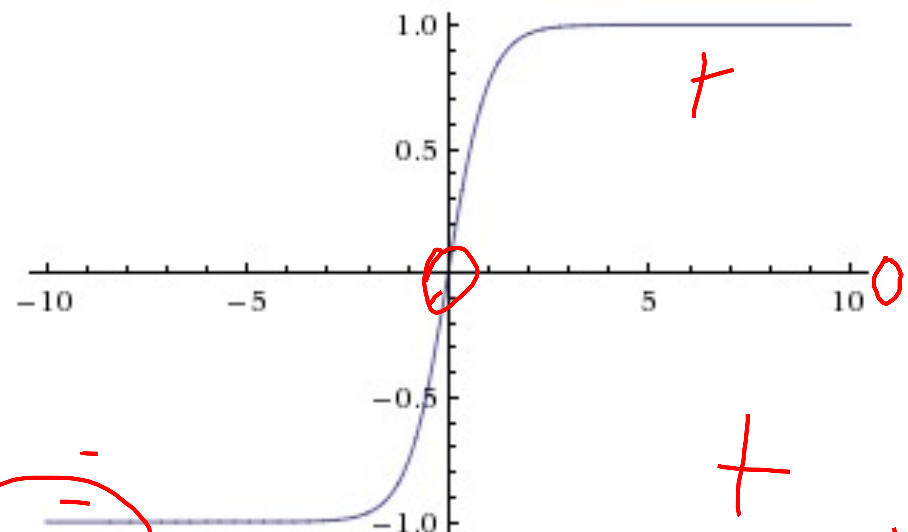
$\sigma(a)$

- sigmoid vs tanh

$\frac{1}{1+e^{-a}}$ $\qquad 2\sigma(2a) - 1$

$a = 10^5$





$\sigma(\Sigma\sigma(\Sigma\sigma(x)))$

A quick note

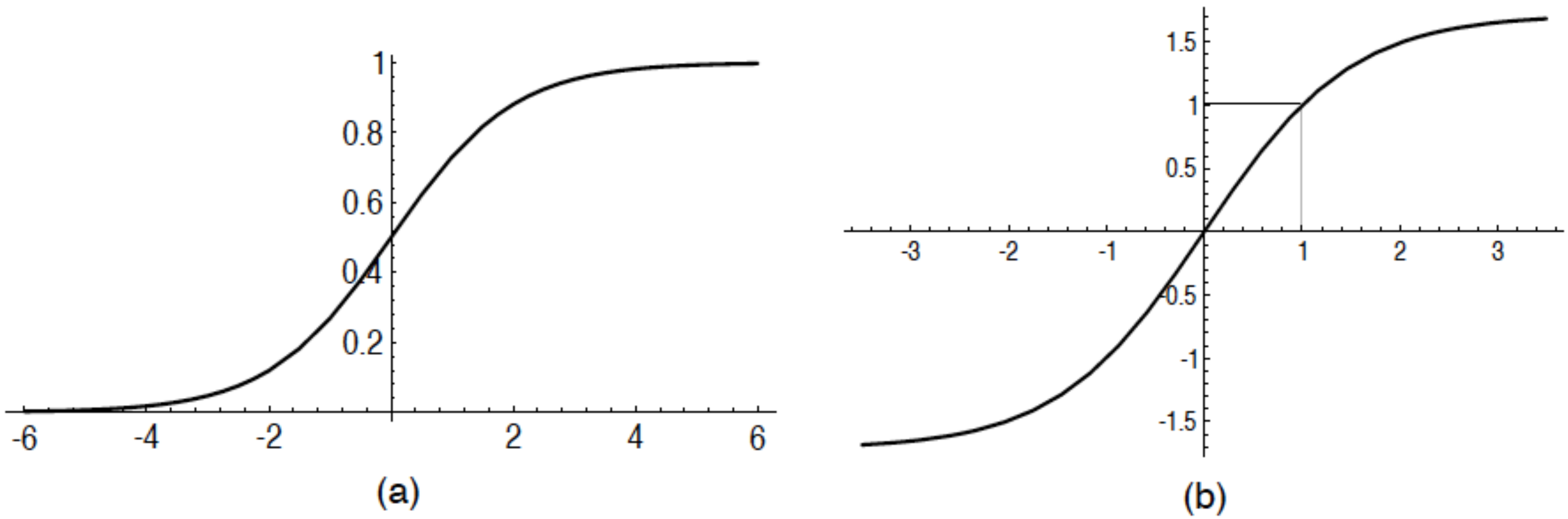
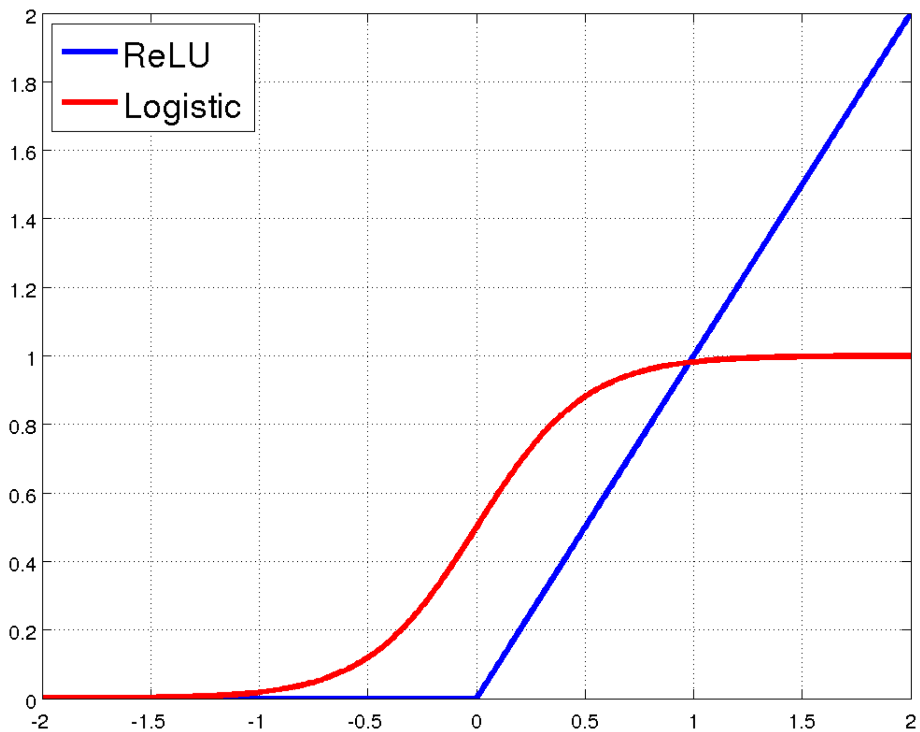
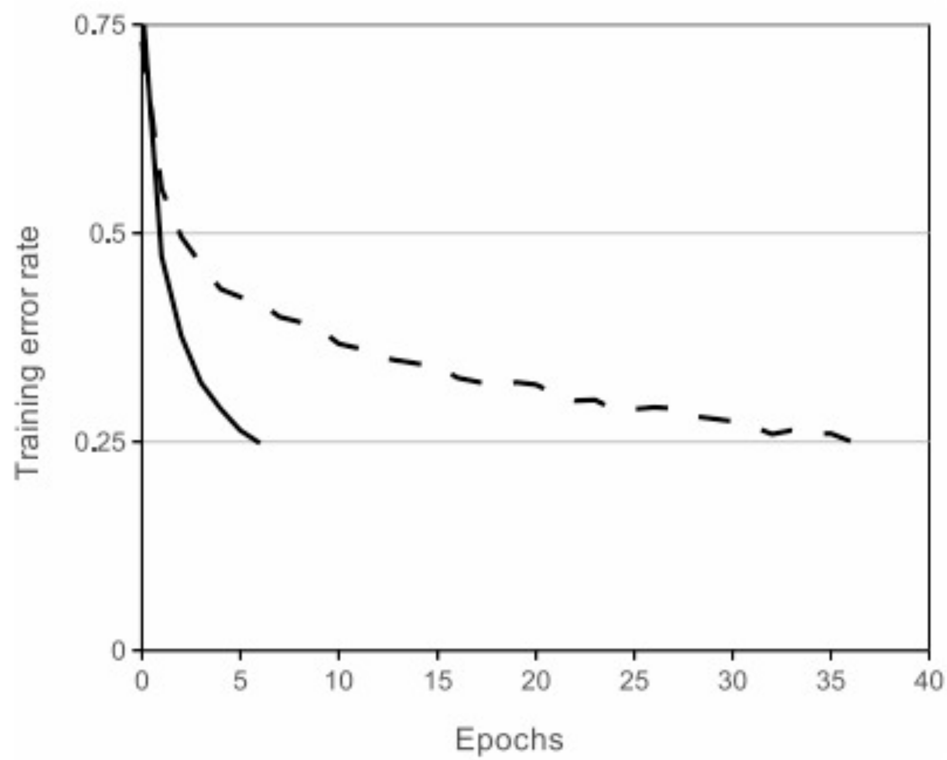


Fig. 4. (a) Not recommended: the standard logistic function, $f(x) = 1/(1 + e^{-x})$. (b) Hyperbolic tangent, $f(x) = 1.7159\ \tanh\left(\frac{2}{3}x\right)$.

Image Credit: LeCun et al. ‘98

Rectified Linear Units (ReLU)





[Krizhevsky et al., NIPS12]

Demo Time

- https://playground.tensorflow.org